(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
15 November 2001 (15.11.2001)

PCT

(10) International Publication Number
WO 01/86626 A2

(51) International Patent Classification[7]: G10H

(21) International Application Number: PCT/GB01/01980

(22) International Filing Date: 4 May 2001 (04.05.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

0010967.8	5 May 2000 (05.05.2000)	GB
0010969.4	5 May 2000 (05.05.2000)	GB
0011178.1	9 May 2000 (09.05.2000)	GB
0022164.8	11 September 2000 (11.09.2000)	GB
0030841.1	18 December 2000 (18.12.2000)	GB

(71) Applicant *(for all designated States except US)*: **SSEYO LIMITED** [GB/GB]; Highview House, Charles Square, Bracknell, Berkshire RG12 1DF (GB).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **COLE, John, Tim** [GB/GB]; Sseyo Limited, Highview House, Charles Square, Bracknell, Berkshire RG12 1DF (GB). **COLE, Murray, Peter** [GB/GB]; Sseyo Limited, Highview House, Charles Square, Bracknell, Berkshire RG12 1DF (GB).

(74) Agents: **MAGGS, Michael, Norman** et al.; Kilburn & Strode, 20 Red Lion Street, London WC1R 4PJ (GB).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

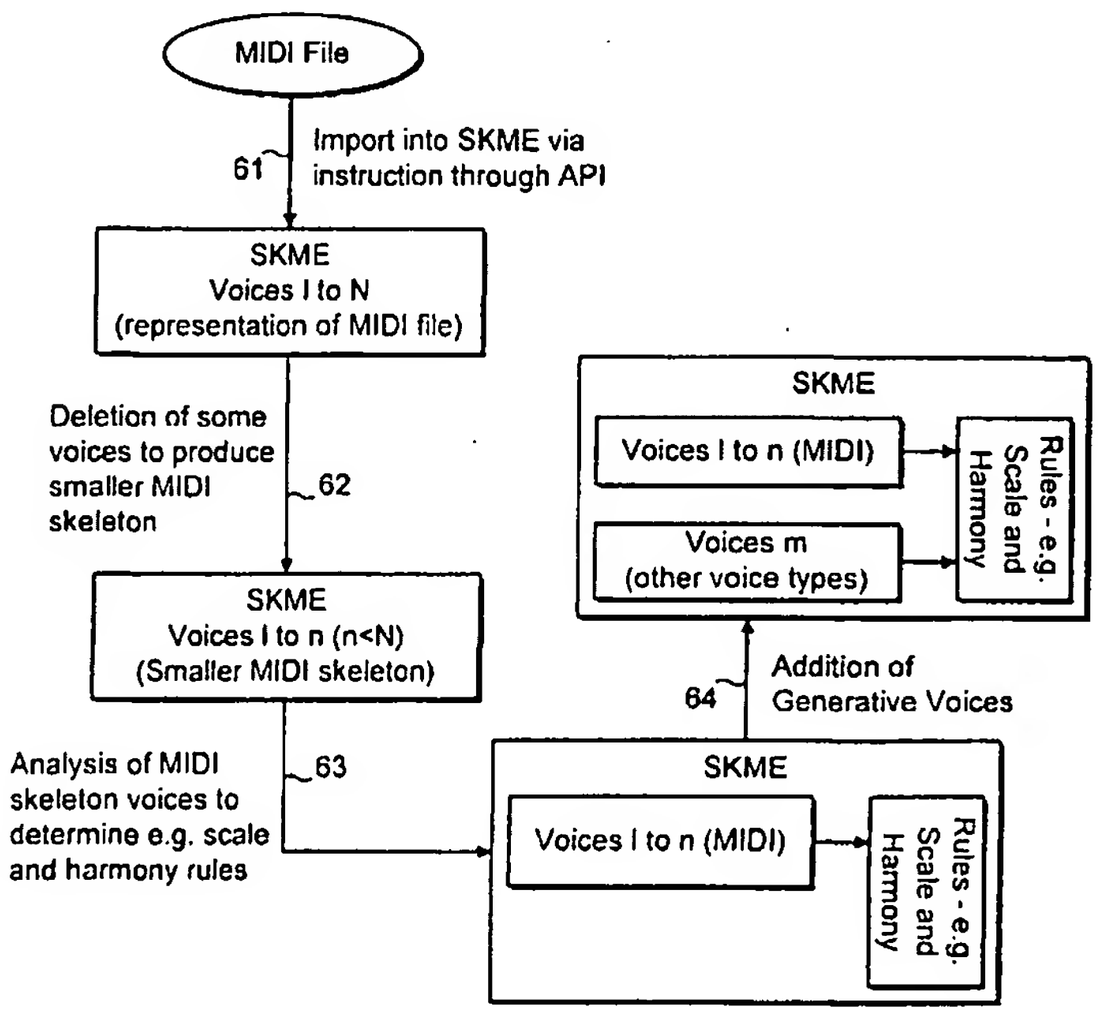
(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report*

[Continued on next page]

(54) Title: AUTOMATED GENERATION OF SOUND SEQUENCES



(57) Abstract: A method of generating a sound sequence comprises deriving a musical skeleton from a MIDI file, and then augmenting that skeleton with music or other sound, to create an automated accompaniment, by means of a generative sound process.

WO 01/86626 A2

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

Automated Generation of Sound Sequences

This invention relates to methods and systems for automated generation of sound sequences, and especially (though not exclusively) of sound sequences in the form of music.

The automated creation of music has a long history, going back at least as far as Mozart's use of musical dice. One of the first musical works generated by a computer was L. Hiller's Illiac suite. Since that time, of course, the sophistication of computer-generated music or more generally audio sequences has increased substantially.

Systems for creating musical sequences by computer may conveniently be divided up into two areas, which have been called "non-generative" and "generative". Non-generative systems include deterministic systems which will produce the same sequences every time, along with systems that simply replay (perhaps in a random or other order) pre-composed sections of music. The vast majority of current systems which produce musical output make use of this type of approach, for example by selecting and playing a particular predefined sequence of notes at random when the key is pressed or a mouse button clicked. Generative Music Systems, on the other hand, may be considerably more complex. Such systems generate musical content, typically note by note, on the basis of a higher-level of musical knowledge. Such systems either explicitly or implicitly are aware of a variety of musical rules which are used to control or influence the generation of the music. In some systems, the rules may operate purely on the individual notes being generated, without imposing any form of higher order musical structure on the output; in such systems, any musical order that arises will be of an emergent nature. More sophisticated systems may include higher-level rules which can influence

the overall musical structure. Generative Music Systems will normally create musical content "on the fly", in other words the musical sequences are built up note by note and phrase by phrase, starting at the beginning and finishing at the end. This means that – in contrast with some of the non-generative systems - the musical content can be generated and played in real time: there is no need for example for the whole of the phrase to be generated before the first few notes of the phrase can be played.

For our present purposes, the essential features of a generative music system are that it generates musical content in a non-deterministic way, based upon a plurality of musical rules (which may either be implicit within the software or which may be explicitly specified by either the program writer or the user of the program). By analogy, a generative sound system produces non-deterministic sound sequences based upon sound-generation rules.

According to an aspect of the present invention there is provided a method of generating a sound sequence comprising deriving a musical skeleton from a music file, and augmenting the skeleton with music or other sound created by a generative sound process.

The invention extends to a generative sound system, and to a computer program, using a method as previously described.

The invention further extends to a generative method of creating music which harmonises with a music file (for example a MIDI file).

According to another of the aspects of the invention, a method or system is provided in which a musical instrument digital interface (MIDI) file, or any other type of parametric music format, or other sounds or audio content is augmented by music or other audio created by a generative sound process.

The invention extends to the augmentation of non-music (e.g. speech) files.

According to another aspect, there is provided a method of generating a sound sequence comprising deriving a skeleton from a file representative of an initial sound sequence, and augmenting the skeleton with music or other sound created by a generative sound process.

The MIDI file may be augmented through the generative sound process with harmonising lines and/or rhythmic accompaniment. Where other audio content is involved, composition may occur in conjunction with rules extracted from the audio.

A method and system for automated generation of sound sequences according to an embodiment of the invention will now be described, by way of example, with reference to the accompanying drawings, in which:

Figure 1 is a schematic representation of the preferred system of the invention;

Figure 2 is illustrative of objects that are involved in a component of the system of Figure 1;

Figure 3 is a flow-chart showing process steps involved in control sequencing within the method and system of the invention;

Figure 4 is illustrative of operation of the method and system of the invention in relation to scale and harmony rules;

Figure 5 illustrates operation of the method and system of the invention in relation to the triggering of note sequences and their integration into a musical work as currently being composed and played; and

Figure 6 illustrates how a MIDI file is augmented in the method and system of the invention.

The method and system to be described are for automated generation of sound sequences and to integrate data presented or interpreted in a musical context for generating an output reflecting this integration. Operation is within the context of generation of musical works, audio, sounds and sound environments in real-time. More especially, the method and system function in the manner of a `generative music system' operating in real-time to enable user-interaction to be incorporated into the composition on-the-fly. The overall construction of the system is shown in Figure 1 and will now be described.

Referring to Figure 1, the system involves four high-level layers, namely, an applications layer I comprising software components 1 to 5, a layer II formed by an application programmer's interface (API) 6 for interfacing with a music engine SKME that is manifest in objects or components 7 to 14 of a layer III, and a hardware device layer IV comprising hardware components 15 to 19 that interact with the music engine SKME of layer III. Information flow between the software and hardware components of layers I to IV is represented in Figure 1 by arrow-heads on dotted-line interconnections, whereas arrow-heads on solid lines indicate an act of creation; for example, information in the composed-notes buffer 11 is used by the conductor 12 which is created by the soundscape 8.

The applications layer I determines the look, feel and physical instantiation of the music engine SKME. Users can interact with the music engine SKME through web applications 1, or through desktop computer applications 2 such as those marketed by the Applicants under their Registered Trade Mark KOAN as KOAN PRO and KOAN X; the music engine SKME may itself be such as marketed by the Applicants under the Registered Trade Mark KOAN. Interaction with the

engine SKME may also be through applications on other diverse platforms 3 such as, for example through mobile telephones or electronic toys. All applications 1 to 3 ultimately communicate with the music engine SKME via the API 6 which protects the internals of the music engine SKME from the outside world and controls the way in which the applications can interact with it. Typically, the instructions sent to the API 6 from the applications 1 to 3 consist of commands that instruct the music engine SKME to carry out certain tasks, for example starting the composition and playback, and changing the settings of certain parameters (which may affect the way in which the music is composed/played). Depending on the needs of the individual applications, communication with the API 6 may be direct or via an intermediate API. In the present case communication to the API 6 is direct from the desktop computer applications 2, whereas it is via an intermediate browser plug-in API 4 and Java API 5 from applications 1 and 3 respectively.

The music engine SKME, which is held in memory within the system, comprises eight main components 7 to 14. Of these, SSFIO 7, which is for file input/output, holds a description of the parameters, rules and their settings used by algorithms within the engine, to compose. When the engine SKME is instructed via the API 6 to start composition/playback, a soundscape 8 is created in memory and this is responsible for creating a composer 10, conductor 12 and all the individual compositional objects 9 relating to the description of the piece as recorded in the SSFIO 7. The compositional objects are referred to by the composer 10 to decide what notes to compose next. The composed notes are stored in a number of buffers 11 along with a time-stamp which specifies when they should be played. The conductor 12 keeps time, by receiving accurate time information from a timer device 19 of level IV. When the current time exceeds the time-stamp of notes in the buffers 11, the relevant notes are removed from the buffers 11 and the information they contain (such as concerning pitch, amplitude, play time, the instrument to be used, etc.) is passed to the appropriate rendering objects 13. The

rendering objects 13 determine *how* to play this information, in particular whether via a MIDI output device 17, or as an audio sample via an audio-out device 18, or via a synthesiser engine 14 which generates complex wave-forms for audio output directly, adding effects as needed.

The hardware devices layer IV includes in addition to the devices 17 to 19, a file system 15 that stores complete descriptions of rules and parameters used for individual compose/playback sessions in the system; each of these descriptions is stored as an `SSfile', and many of these files may be stored by the file system 15.

In addition, a MIDI in device 16 is included in layer IV to allow note and other musical-event information triggered by an external hardware object (such as a musical keyboard) to be passed into the music engine SKME and influence the composition in progress.

The system can be described as having essentially two operative states, one, a `dynamic' state, in which it is composing and the other, a `static' state, in which it is not composing. In the static state the system allows modification of the rules that are used by the algorithms to later compose and play music, and keeps a record encapsulated in the SSFIO component 7, of various objects that are pertinent to the description of how the system may compose musical works. The system is also operative in the dynamic state to keeps records of extra objects which hold information pertinent to the real-time composition and generation of these works. Many of these objects (the compositional objects 9 for example) are actual instantiations in memory of the descriptions contained in the SSFIO 7. Modification of the descriptions in the SSFIO 7 via the API layer II during the dynamic state, results in those modifications being passed down to the compositional objects 9 so that the real-time composition changes accordingly.

Figure 2 shows a breakdown of the SSFIO component 7 into its constituent component objects which exist when the system is in its static and dynamic

states; the system creates real-time versions of these objects when composing and playing. In this respect, SSfiles 20 stored each provide information as to `SSObject(s)' 21 representing the different *types* of object that can be present in the description of a work; these objects may, for example, relate to piece, voice, scale rule, harmony rule, rhythm rule. Each of these objects has a list of `SSFparameters' 22 that describe it; for example, they may relate to tempo, instrument and scale root. When an SSfile 20 is loaded into the music engine SKME, actual instances of these objects 21 and their parameters 22 are created giving rise to `SSFObjectInstance' 23 and `SSFParameterInstance' 24 as illustrated in Figure 2.

Referring again to Figure 1, the user interacts with the system through applications 1 to 3 utilising the services of the API 6. The API 6 allows a number of functions to be effected such as `start composing and playing', `change the rules used in the composition', `change the parameters that control how the piece is played' including the configuration of effects etc. One of the important aspects of the method and system of the invention is the ability to trigger generative pattern sequences in response to external events. The triggering of a generative pattern sequence has a range of possible outcomes that are defined by the pattern sequence itself. In the event that a generative pattern sequence is already in operation when another trigger event is received, the currently operational sequence is ended and the new one scheduled to start at the nearest availability.

Generative pattern sequences allow a variety of musical seed phrases of any length to be used in a piece, around which the music engine SKME can compose in real time as illustrated in Figure 3. More particularly, the generative pattern sequence contains a collection of one or more note-control sub-patterns with or without one or more additional sequence-control sub-patterns. Three types of note-control sub-patterns can be created, namely: `rhythm' note-control sub-pattern containing note duration information, but not assigning specific

frequencies to use for each note; `frequency and rhythm' note-control sub-pattern containing both note duration and some guidance to the generative music engine SKME as to the frequency to use for each note; and `forced frequency' note-control sub-pattern containing note duration, temporal positioning and explicit frequency information to use for each note. Sequence-control sub-patterns, on the other hand, can be used to specify the sequence in which the note-control sub-patterns are played, and each note-control sub-pattern may also specify ranges of velocities and other musical information to be used in playing each note. The music engine SKME allows the use of multiple sub-patterns in any generative pattern sequence.

Referring to Figure 3, the step 30 of triggering the generative pattern sequence acts through step 31 to determine whether there are any other sequence-control sub-patterns operative. If not, a note-control sub-pattern is chosen at random in step 32 from a defined set; each note-control sub-pattern of this set may be assigned a value that determines its relative probability of being chosen. Once it is determined in step 33 that the selected note-control sub-pattern is finished, another (or the same) note-control sub-pattern is selected similarly from the set. The generative pattern sequence continues to play in this manner until instructed otherwise.

If the result of step 31 indicates that there is one or more sequence-control sub-patterns operative, then any sequence-control sub-pattern is chosen at random in step 34 from the defined set; each sequence-control sub-pattern may be assigned a value that determines its relative probability of being chosen. Once a sequence-control sub-pattern has been selected in step 34, it is consulted to determine in step 35 a sequence of one or more note-control sub-patterns to play. As each note-control sub-pattern comes to an end, step 36 prompts a decision in step 37 as to whether each and every specified note-control sub-pattern of the operative sequence has played for the appropriate number of times. If the answer is NO,

then the next note-control sub-pattern is brought into operation through step 35, whereas if the answer is YES another, or the same, sequence-control sub-pattern is selected through repetition of step 34. As before, the generative pattern sequence continues to play in this manner until instructed otherwise.

Each sequence-control sub-pattern defines the note-control sub-pattern(s) to be selected in an ordered list, where each entry in the list is given a combination of: (a) a specific note-control sub-pattern to play, or a range of note-control sub-patterns from which the one to play is chosen according to a relative probability weighting; and (b) a value which defines the number of times to repeat the selected note-control sub-pattern, before the next sequence-control sub-pattern is selected. The number of repetitions may be defined as a fixed value (e.g. 1), as a range of values (e.g. repeat between 2 and 5 times), or as a special value indicating that the specified note-control sub-pattern should be repeated continuously.

Depending upon the note-control sub-pattern operational at any moment after a generative pattern sequence is triggered, various rules internal to the music engine SKME may be used to determine the exact pitch, duration and temporal position of the notes to be played. For example, if a `rhythm' note-control sub-pattern is in operation at a particular point in the generative pattern sequence, then the scale rule, harmony rule and next-note rule within the music engine SKME for that `triggered voice' will be consulted to obtain the exact notes. Alternatively, if the `forced frequency' note-control sub-pattern is operational, no internal rules need be consulted since all the note information is already specified. Furthermore, for the case of `frequency and rhythm', the music engine SKME combines the given frequency offset information with its rules and other critical information such as the root of the current scale and range of available pitch values for the voice in question.

The rules and other parameters affecting composition (e.g. tempo) within the music engine SKME are defined in memory, specifically within the SSFIO 7, and its real-time instantiation of the compositional objects 9. Use of rules and parameters within the music engine SKME form part of the continual compositional process for other voice objects within the system. Figure 4 illustrates this more general process based on examples of scale and harmony rules shown at (1) and (2) respectively.

Referring to Figure 4, the scale rule is illustrated at (1) with shaded blocks indicating a non-zero probability of choosing that interval offset from a designated scale root note. The larger the shaded block, the greater the probability of the system choosing that offset. Thus, for this example, the octave Ove, major third M3 and fifth 5 are the most likely choices, followed by M2, 4, M6 and M7; the rest will never be chosen. Sequences that may be generated by the system from this are shown below the blocks, and in this respect the octave has been chosen most often followed by the major third and the fifth. With the scale root set in the system as C, the resulting sequence of notes output from the system in this example are C,E,C,D,G,A,E,D,C,G,E,B,C,F, as illustrated at (1) of Figure 4.

The harmony rule defines how the system may choose the pitches of notes when other notes are playing, that is to say, how those pitches should harmonise together. In the example illustrated at (2) of Figure 4, only the octave and major second are indicated (by shading) to be selected. This means that when the pitch for a voice is chosen, it must be either the same pitch as, or a major second from, all other notes currently being played.

For the purpose of further explanation, consideration will be given to the example represented at (3) of Figure 4 involving three voice objects V1-V3. The rhythm rules applicable to the voice objects V1-V3 in this example, give rise to a

generated sequence of notes as follows: voice V1 starts playing a note, then voice V2 starts playing a note, then voice V3 starts playing a note, and then after all notes have ended, voice V2 starts playing another note, followed by voice V1 and then voice V3. With this scenario, the note from voice V2 must harmonise with that of voice V1 and the voice V3 note must harmonise with that of voice V2. If in these circumstances the voice V1 is, as illustrated by bold hatching, chosen with a pitch offset of a fifth from the scale root, the pitch for voice V2 must either be the same as (Ove) or a major second above (M2) the fifth. In the case illustrated, it is chosen to be the same, and so the fifth is chosen too. When voice V3 starts playing it must harmonise with both voices V1 and V2, so the pitch chosen must be the same as, or a major second above that of voices V1 and V2. As illustrated, the system chooses voice V3 to be a major second above, therefore giving pitch offset M6 from the scale root.

After voice V3 all notes end, and the next note begins, as illustrated at (4) of Figure 4 with voice V2. This next note by voice V2 is governed by the next-note rule used by voice V2, and the last note played by voice V2. According to this rule, the system chooses pitch offset M2 for voice V2, and then harmonises voices V3 and V1 with it by choice of a major second for both of them. With the scale root set in the system to C, the entire generated sequence accordingly follows that indicated at (5) of Figure 4, where `S' denotes a note starting and `E' a note ending.

Thus, when sequences are generated in response to an external trigger, the actual pitches and harmonisation of that sequence is determined by the composer 10 using several items of information, namely: (a) the note-control sub-pattern operational at that moment; (b) the scale, rhythm, harmony and next-note rules depending upon the type of the note-control subsequence; and (c) any piece-level rules which take into account the behaviour of other voices within the piece.

When the music engine SKME is in dynamic (i.e. composing and playing) mode, it typically contains a number of voice compositional objects 9. The composer 12 composes a sequence of notes for each of these and makes sure they obey the various rules. The process involved is illustrated in the flow diagram of Figure 5.

Referring to Figure 5, the music engine SKME responds to a trigger applied at step 51, and the API 6 through step 52 instructs a voice 1 in step 53 to register that it must start a sequence. Voice 1 and the voices 2 to N in step 54, have their own rules, and the composer 10 ensures that the relevant rules are obeyed when utilising any of the voices 1 to N. More particularly, the composer 10 responds in step 55 to the instruction of step 53 for voice 1 to start a sequence, by starting the generative pattern sequence sub-system of Figure 3. This sends note-control sub-sequences to the trigger voice (voice 1 in this example), but the composer 10 makes sure the resulting notes harmonise with the other voices in the piece. The outcome via the conductor 12 in step 56 is played in step 57.

The generative pattern sequence triggered will play forever, or until the system is instructed otherwise. If a sequence control sub-pattern is used to define a generative pattern sequence such that the final note control sub-pattern is one which plays silence (rest notes) in an infinite loop, then when this pattern sequence is selected, the voice will become effectively `inactive' until another trigger is detected. Further triggering events for the same generative pattern sequence may sound different as the process is generative, or since the rules in use by the piece or the scale of the trigger voice, its harmony or next note rules may have changed (either via interaction through the API 6 or via internal music engine SKME changes).

The sounds used to `render' each note, whether from triggered sequences or generative voices may be played either through the MIDI sounds or the samples of the rendering objects 13, or via software of the synthesiser engine 14 which

may add digital signal processing effects such as, for example, filter sweeps, reverberation and chorus. The entire process can be used to generate musical event information that is then fed into, and may thus control, other processing units within the system such as synthesiser related units allowing the triggering of generative sound effects. Voices can also be added which make use of the software synthesiser engine 14 to generate non note-based effects such as sound washes and ambient environmental sounds, such as chimes, wind and other organic sounds.

The system provides the ability to augment the playback of complete or partial MIDI files (or other pre-defined music files) with harmonising musical lines and effects. Figure 6 illustrates the process involved, with specific reference to MIDI. In this a MIDI track imported at step 61 into the music engine SKME via the API 6 is split up into a number of voices 1 to N, each typically representing a different instrumental part; if a single part is complex, it may itself be split up into a number of voices. As indicated in step 62, some of these voices may, if desired by the user, be deleted, to leave a less complex framework or skeleton version (voices 1 to n) of the original MIDI file. The remaining voices 1 to n may be used by the composer 10 to produce a faithful rendition of the MIDI file, or alternatively, the user may via the API 6, supplement the MIDI voices with other voice types used within the generative system or those that track the MIDI voices themselves, i.e. those that create random parts which follow the scale, rhythm and harmony rules of the system.

Many options are possible. The user may wish, for instance, to mute out all the lines in the MIDI file except the bass, and to re-generate a whole new piece simply based upon that line. A line which has been muted may be allowed to influence generation of the music, even though it cannot itself be heard. Alternatively, individual lines may be deleted in their entirety, in which case they will have no effect on the musical generation at all. The music generation

system may be instructed to leave some lines in the MIDI file untouched, and to generate music around them. The system may also provide the flexibility for the user to mute or delete individual notes. The user may also wish to take the note values alone (ignoring the note lengths) and to ask the system to change the rhythm (e.g. to play in 6/8 rather than 3/8).

For maximum correlation between the generative voices and the MIDI derived Voices, the MIDI file is analysed in step 63 to extract the relative probabilities of occurrence of different pitches. This is converted into scale and harmony rules which may be applied to one or more of the generative voices, ensuring that the generative accompaniment added in step 64 does not sound discordant when heard with the MIDI voices.

If the system, for example, determines that C is the most frequently-used note, it may conclude that the root note of the key to use is C. The system might also conclude, for example, that the piece should be rendered in the key of C Major, different relative probabilities being accorded to each note within the scale. Other information might also be extracted, including (but note limited to) information on the harmonies and rhythmic structures used within the pre-defined music file.

The system could also be arranged to avoid those notes which it determines have not been used within the original MIDI file.

The file may be downloaded and worked on as one unit, or it could be worked on as it arrives (e.g. streaming). The file could be a (rendered) audio file from which note information or event information could be extracted for use as the skeleton. The file need not be representative of music: it could for example be a non-music file (e.g. speech) from which may be extracted note information or event information which is capable of being converted into note or rhythm

information for use as the skeleton.

The preferred system incorporates an easy to use user interface, allowing the user to modify a variety of settings and to receive immediate feedback. The user does not need to be a programmer or a composer, but simply creates music based on the framework of the original MIDI file in a very interactive way. The user can make changes very easily, and listen immediately to what those sound like. Gradually, the user refines the settings to create a work that he or she is happy with.

As with the earlier-described feature, the sounds used to 'render' each note, whether from MIDI or generative voices may be played either through the MIDI sounds or the samples of the rendering objects 13, or via software of the synthesiser engine 14 which may add digital signal processing effects such as, for example, filter sweeps, reverberation and chorus. In addition to adding generative tracking voices, voices can be added which make use of the software synthesiser engine 14 to generate non note-based effects such as sound washes and ambient environmental sounds, such as chimes, wind and other, organic sounds.

CLAIMS:

1. A method of generating a sound sequence comprising deriving a musical skeleton from a music file, and augmenting the skeleton with music or other sound created by a generative sound process.

2. A method of generating a sound sequence as claimed in claim 1 in which the music file is a MIDI file.

3. A method of generating a sound sequence as claimed in claim 1 or claim 2 in which the generative sound process generates musical notes which harmonise with the musical skeleton.

4. A method of generating a sound sequence as claimed in any one of claims 1 to 3 in which the music file includes a plurality of musical voices, the musical skeleton being derived by muting or deleting some of the musical voices.

5. A method of generating a sound sequence as claimed in claim 4 in which the generative sound process generates music or other sound at least partially in dependence upon one or more muted voices.

6. A method of generating a sound sequence as claimed in any one of the preceding claims in which the music file includes a key indicator, the generative sound process generating music in a key specified by the key indicator.

7. A method of generating a sound sequence as claimed in any one of claims 1 to 5 in which the generative sound process attempts to determine the key used by the music file.

8. A method of generating a sound sequence as claimed in claim 7 in which

the key is determined by creating a histogram of the notes contained within the music file.

9. A method of generating a sound sequence as claimed in any one of the preceding claims in which the generative music process has a user-option allowing the rhythm of the skeleton to be changed.

10. A method of generating a sound sequence as claimed in any one of the preceding claims in which the generative sound process avoids the use of musical notes which are not present in the music file.

11. A method of generating a sound sequence as claimed in any one of claims 1 to 9 in which the generative sound process avoids the use of musical notes which are not present in the skeleton.

12. A generative sound system using a method as claimed in any one of the preceding claims.

13. A computer program using a method as claimed in any one of the preceding claims.

14. A method of generating a sound sequence as claimed in claim 1 in which the skeleton includes harmonic information derived from the music file.

15. A method of generating a sound sequence as claimed in claim 1 in which the skeleton includes rhythmic information derived from the music file.

16. A method of generating a sound sequence as claimed in claim 14 including generating musical notes which reflect the harmonic information.

17. A method of generating a sound sequence as claimed in claim 15 including generating musical notes which reflect the rhythmic information.

18. A method of generating a sound sequence comprising deriving a skeleton from a file representative of an initial sound sequence, and augmenting the skeleton with music or other sound created by a generative sound process.

19. A method of generating a sound sequence as claimed in claim 19 in which the file is representative of speech.

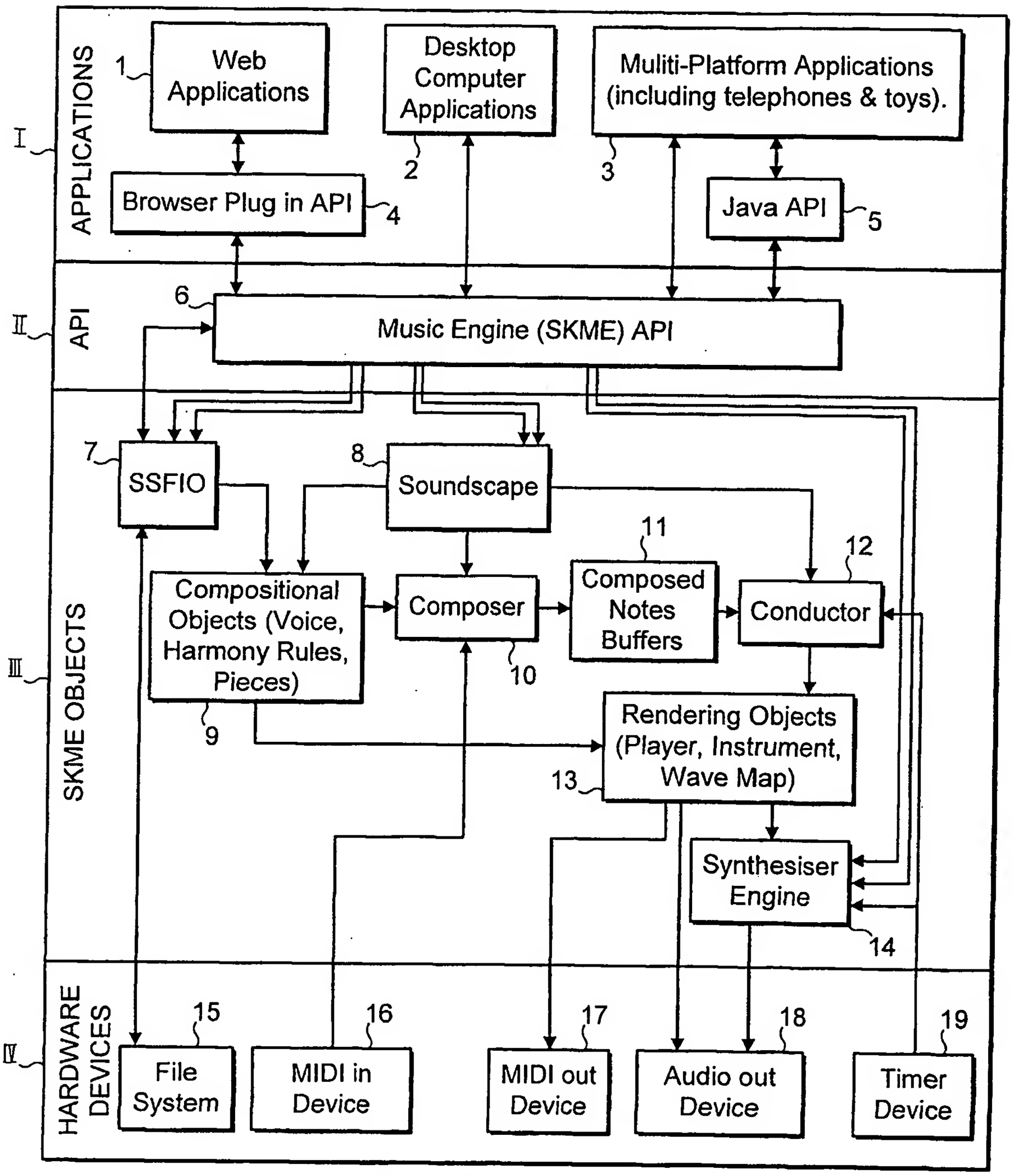

I
APPLICATIONS
1
Web Applications
Desktop Computer Applications
2
Muliti-Platform Applications (including telephones & toys).
3
Browser Plug in API
4
Java API
5
II
API
6
Music Engine (SKME) API
III
SKME OBJECTS
7
SSFIO
8
Soundscape
11
12
Compositional Objects (Voice, Harmony Rules, Pieces)
9
Composer
10
Composed Notes Buffers
Conductor
Rendering Objects (Player, Instrument, Wave Map)
13
Synthesiser Engine
14
IV
HARDWARE DEVICES
15
File System
16
MIDI in Device
17
MIDI out Device
18
Audio out Device
19
Timer Device

FIG. 1

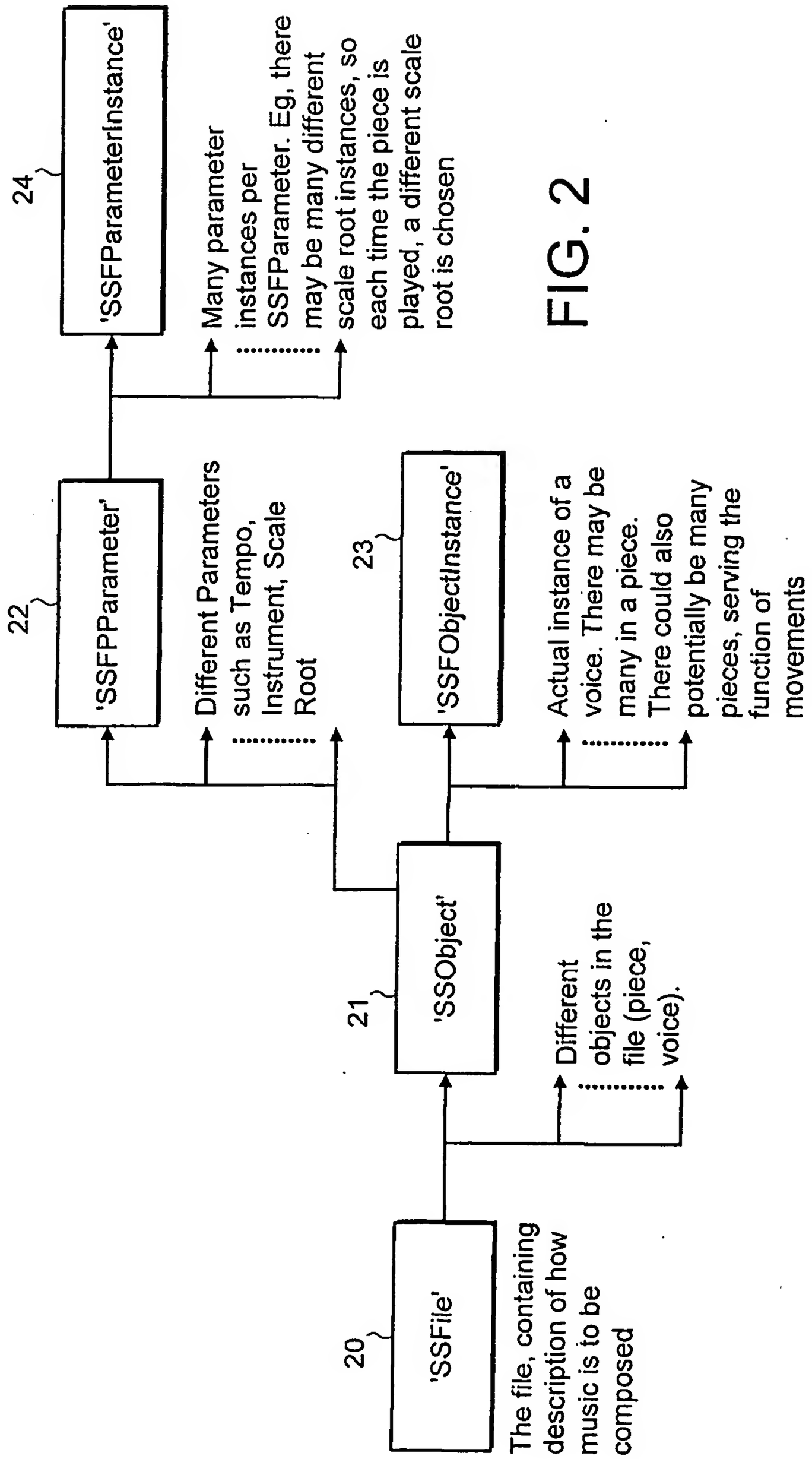

'SSFile'
20
The file, containing description of how music is to be composed
'SSObject'
21
Different objects in the file (piece, voice).
'SSFParameter'
22
Different Parameters such as Tempo, Instrument, Scale Root
'SSFObjectInstance'
23
Actual instance of a voice. There may be many in a piece. There could also potentially be many pieces, serving the function of movements
'SSFParameterInstance'
24
Many parameter instances per SSFParameter. Eg, there may be many different scale root instances, so each time the piece is played, a different scale root is chosen

FIG. 2

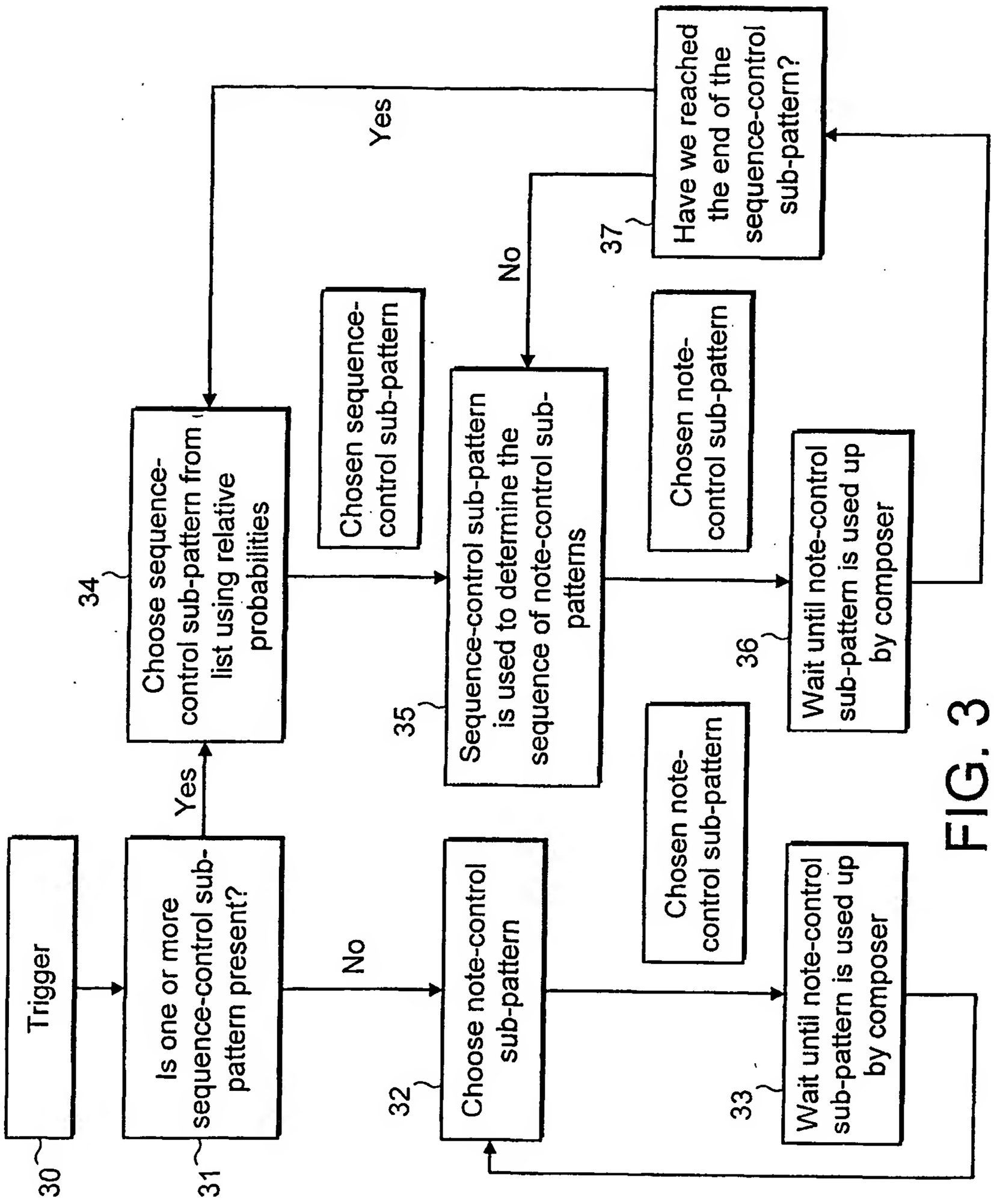
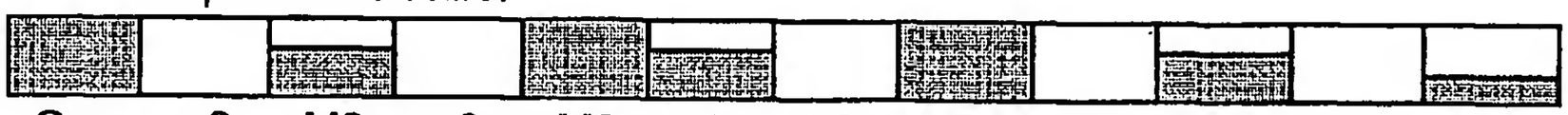

Trigger
30
Is one or more sequence-control sub-pattern present?
31
Yes
No
Choose note-control sub-pattern
32
Wait until note-control sub-pattern is used up by composer
33
Chosen note-control sub-pattern
Choose sequence-control sub-pattern from list using relative probabilities
34
Chosen sequence-control sub-pattern
Sequence-control sub-pattern is used to determine the sequence of note-control sub-patterns
35
Wait until note-control sub-pattern is used up by composer
36
Chosen note-control sub-pattern
Have we reached the end of the sequence-control sub-pattern?
37
Yes
No

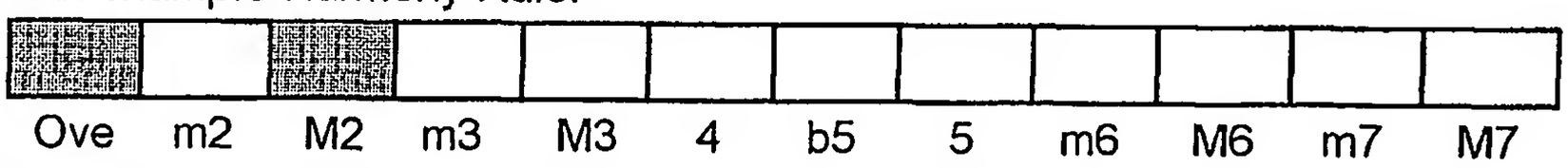
FIG. 3

1 An example Scale Rule:

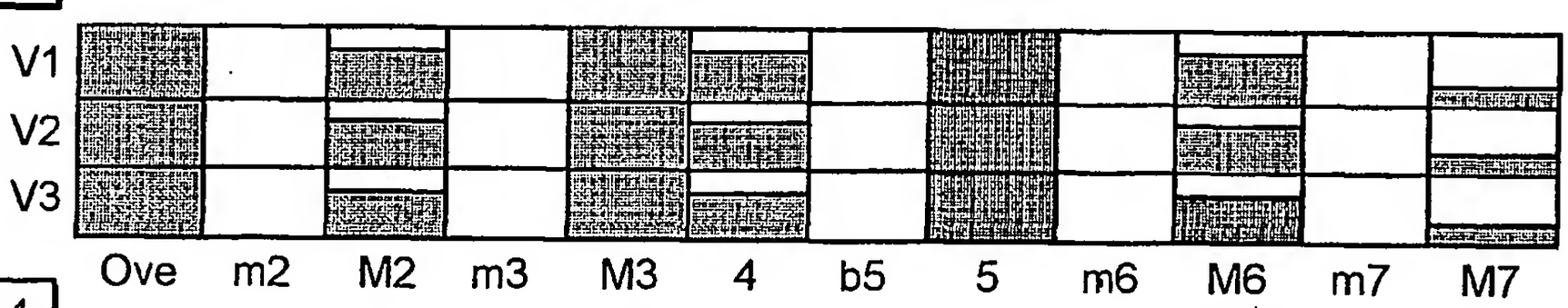


Relative to a scale root, a sequence of notes chosen from this scale rule might be:
Ove, M3, Ove, M2, 5, M6, M3, M2, Ove, 5, M3, M7, Ove, 4
With the scale root set at C, this sequence becomes:
C, E, C, D, G, A, E, D, C, G, E, B, C, F

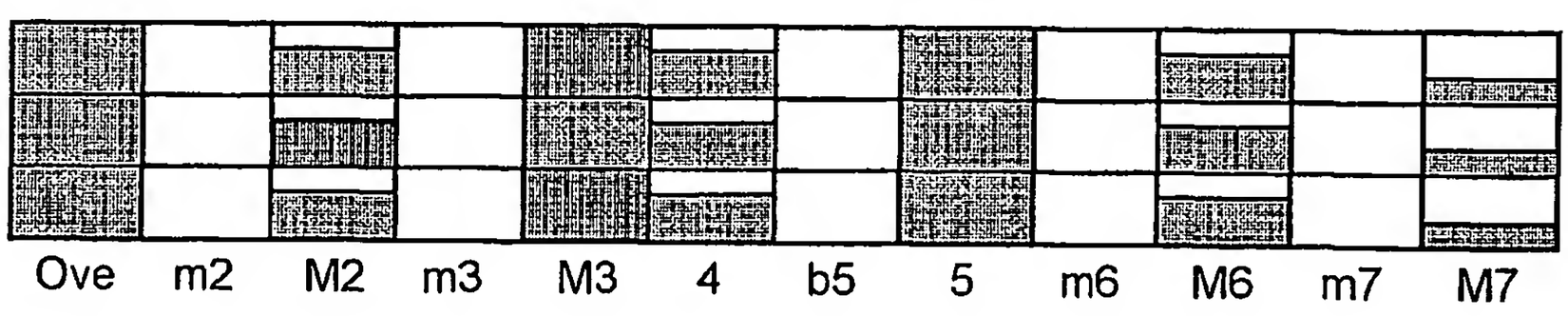
2 An example Harmony Rule:



3



4



5

Time Sequence:
S:V1:G, S:V2:G, S:V3:A,
E:V1. E:V2, E:V3
S:V2:D, S:V3:E, S:V1:E
E:V2, E:V3, E:V1

FIG. 4

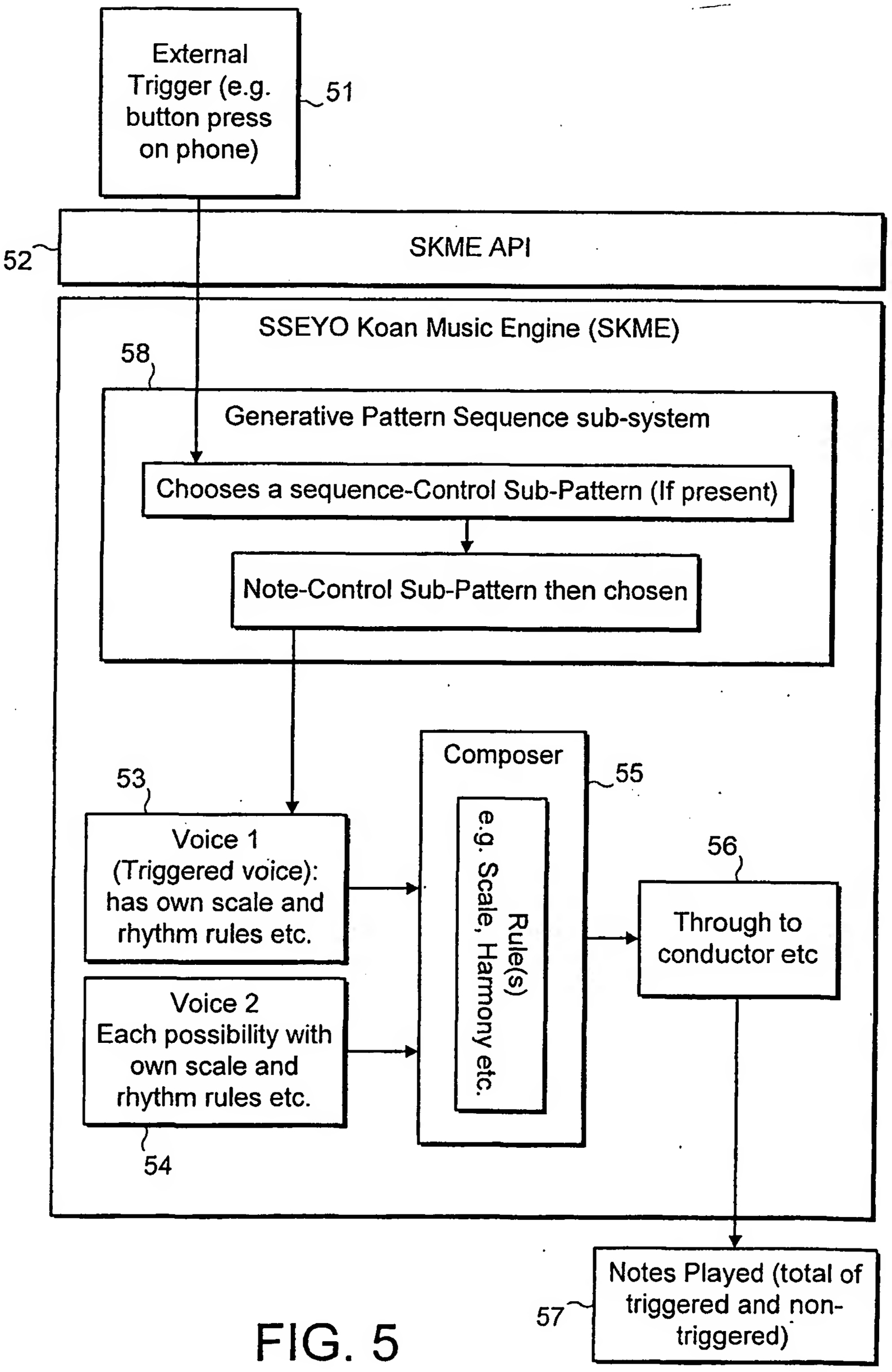

External Trigger (e.g. button press on phone)
51
52
SKME API
SSEYO Koan Music Engine (SKME)
58
Generative Pattern Sequence sub-system
Chooses a sequence-Control Sub-Pattern (If present)
Note-Control Sub-Pattern then chosen
Composer
55
53
Voice 1 (Triggered voice): has own scale and rhythm rules etc.
Rule(s) e.g. Scale, Harmony etc.
56
Through to conductor etc
Voice 2 Each possibility with own scale and rhythm rules etc.
54
57
Notes Played (total of triggered and non-triggered)

FIG. 5

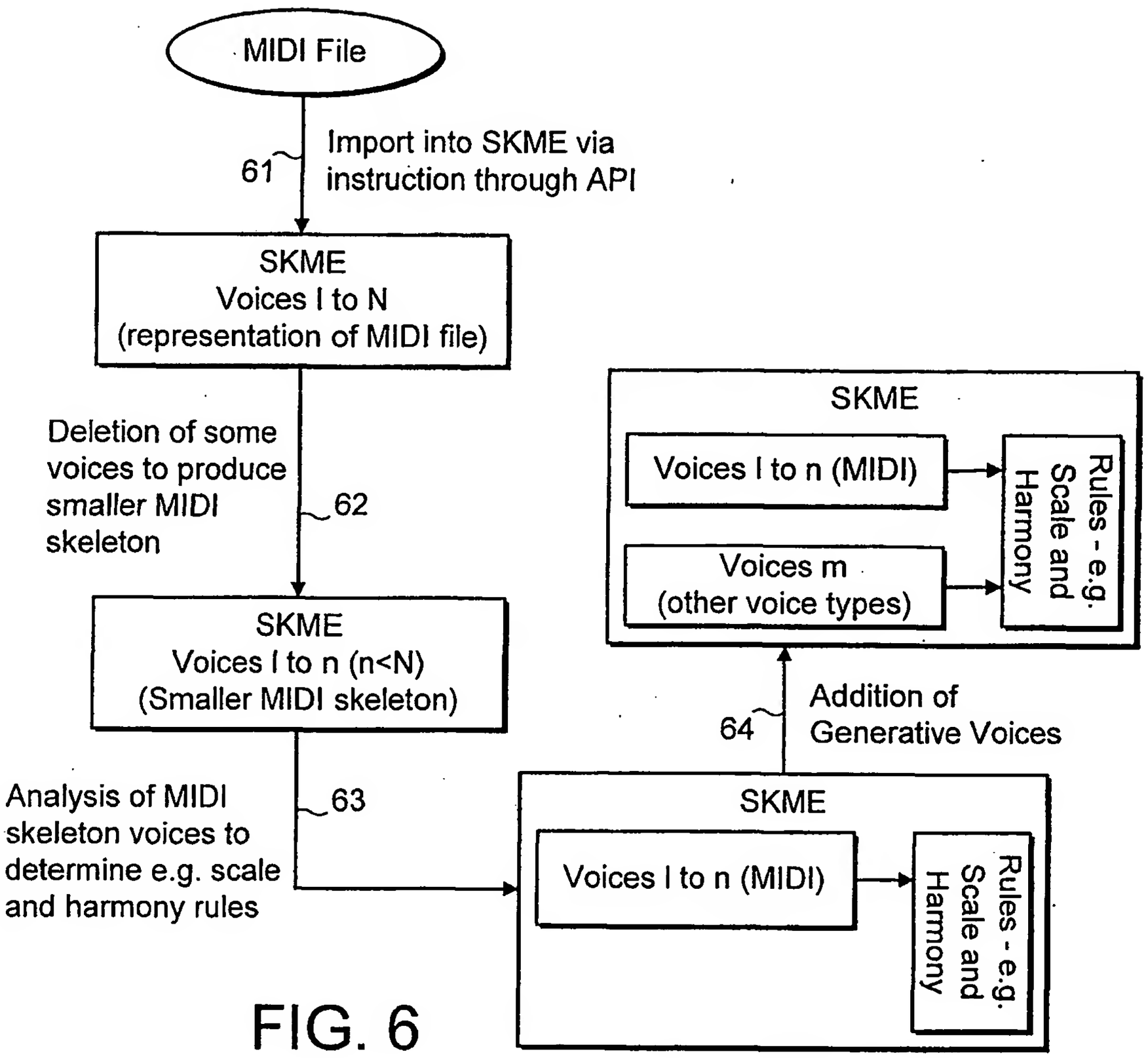

MIDI File
61
Import into SKME via instruction through API
SKME
Voices I to N
(representation of MIDI file)
Deletion of some voices to produce smaller MIDI skeleton
62
SKME
Voices I to n (n<N)
(Smaller MIDI skeleton)
Analysis of MIDI skeleton voices to determine e.g. scale and harmony rules
63
SKME
Voices I to n (MIDI)
Voices m
(other voice types)
Rules - e.g. Scale and Harmony
64
Addition of Generative Voices
SKME
Voices I to n (MIDI)
Rules - e.g. Scale and Harmony

FIG. 6